APPENDIX

Principles of genetic algorithm search and optimization

An important characteristic feature of genetic algorithms is that they operate on a coding (*e.g.*, a binary coding) of the parameters rather than on the parameters themselves [77,104-108]. Thus, the first task of a genetic algorithm is to generate a set of random numbers in a particular encoding that corresponds to the variables of the problem. Each variable is called a "gene" or "allele" and represents a particular feature or character such as, for instance, the thickness or refractive index of a layer in a diffractive optics problem. By combining several genes one obtains a string called a "chromosome," which represents a candidate solution. For example, in a homogeneous-layer thin-film optimization procedure a chromosome would be composed of the thicknesses and the refractive indices of all the layers in a structure. Many such candidate solutions are generated simultaneously by a genetic algorithm. Together they form a "population," and successive populations generated by the genetic algorithm are referred to as "generations."

There are many implementations of genetic algorithms, but they all have in common the basic operators of selection, crossover, and mutation [77,104-108]. All these operators are applied on the population of a generation to create the next generation. A typical flow chart for a genetic algorithm optimization procedure is presented in **FIG. 24**. An initial population is generated randomly with each gene spanning its allowed range of values. The domain can be discrete for some genes and continuous for others, the only restriction consisting in the locations of the genes in the chromosome, which must remain the same for all chromosomes. A merit (sometimes referred to as cost, fitness, or residual) function is calculated for each chromosome. This merit function is problem specific and the success of the optimization procedure depends largely on the choice of the merit function. The chromosomes are ranked in terms of their performance evaluated by the merit function. Successive generations are then created by retaining a part of the chromosomes from one generation to the next and by forming new chromosomes through recombination of the best chromosomes in the old population. The greater the fitness value of a chromosome, the more likely it is to participate in the recombination process. Some algorithms retain a fixed number of

chromosomes [110] while others are more problem-specific and retain all chromosomes with fitness better than a user-defined value [111].

The recombination process consists of applying either or both crossover and mutation operators. In the crossover operation, two chromosomes exchange portions of their encoded representation. The three types of crossover mechanisms encountered in genetic algorithms are illustrated in **FIG. 25** [112]. A single-point crossover is realized by choosing a point in the chromosome chains at random and exchanging the data to the right of this point between the parent chromosomes. In the two-point crossover the data between two randomly selected points is swapped while in the multiple-point crossover data is exchanged at random between the two parent chromosomes. Higher ranked strings will be more likely to participate in the crossover and thus form new chromosomes.

The crossover operation is a random but structured information exchange between chromosomes and represents the essential tool in local searches, (*i.e.*, in exploring points within the hyperplanes already represented in the population) [105]. However, crossover alone would produce convergence in local extrema and, to explore other points in space and avoid "premature convergence," the mutation operator is introduced in the genetic algorithm.

A mutation is the random change of a gene from one value to another. Mutation is carried out with a user-defined probability and according to the statistical rules implemented in the genetic algorithm program. Mutation has a very important role in the search process ensuring variability in the population and hence, avoiding the entrapment of the algorithm in local extrema of the merit function.

The operators of selection, crossover and mutation are independent of the application, and only the merit function contains domain-specific knowledge. Another operator utilized in some genetic algorithms to diversify the search is the restarting operator. After a number of generations the best string is retained while all the others are discarded, and the whole population is reseeded as mutant variations of the best string. For the same purpose of avoiding premature convergence, other genetic operators

introduce a random disturbance in every chromosome of a population after a number of generations or when all chromosomes have reached the same set of genes.

The genetic algorithm ends after a user-determined fixed number of iterations, when the merit function has reached an extremum that is close enough to the desired value, or when all chromosomes in a population have merit functions within a small enough range.

Genetic Algorithm Program for Multilayer Waveguide Gratings

Program description

A genetic algorithm program has been developed for optimization of diffractive optics structures with multiple homogeneous and grating layers and incident TE polarized plane waves [65,71,77]. The program employs rigorous coupled-wave analysis for calculation of the reflected and transmitted diffraction efficiencies [84-86] and, hence, for evaluation of the merit function for the generated structures. The software library PGAPack [110] performs specific genetic algorithm operations (chromosome generation, ranking, selection, crossover, mutation, *etc.*)

The algorithm seeks to find the physical parameters of the diffractive structure that generates the spectral dependence of the zero-order reflected (or transmitted) diffraction efficiency provided by the user in a reference data file. Alternatively, the optimization can be performed for the angular dependence of the diffraction efficiency and at a fixed wavelength of the incident light. The physical parameters to be found in the optimization process are the grating period, the refractive indices, and the thicknesses of the layers, the fill factors, and relative spatial phase shifts of the gratings. The refractive indices of the cover and substrate, the angle of incidence (or the wavelength for angular dependence optimization), the maximum number of layers, and the minimum and maximum values for the thicknesses, fill factors, and grating period are required as input parameters to the program. The refractive indices of the candidate solutions are selected from a list of discrete values supplied by the user in a separate input file. All other physical parameters (grating period, fill factors, thicknesses) are allowed to vary

continuously within the ranges established by the user. Therefore, the program seeks the minimum of the merit function in a mixed discrete and continuous parameter space. This is a practical approach since in fabrication of diffractive optical structures only a limited number of materials can be used in a given spectral range while the thicknesses, fill factors, and the grating period can be varied continuously within a range, and within the accuracy limitations of the equipment. In some applications one or more of the physical parameters may be fixed due to either fabrication constraints (*e.g.*, fill factor of the grating equal to 0.5) or user knowledge about the physics of the problem (*e.g.*, known grating period for center wavelength of resonance filters). This feature is included in the program and will expedite the search procedure by reducing the dimension of the parameter space. A priori information can also simplify the search procedure by reducing the range over which a parameter can vary during optimization, thereby reducing the total number of points in the parameter space.

The physical parameters of the diffractive structure can be encoded as binary, binary Gray, or real numbers [110]. For binary and binary Gray representations of real and integer numbers, the user must specify the number of bits for encoding the thicknesses, the grating period, the fill factors, and the refractive indices. The number of bits allocated for each variable determines the accuracy of the representation of real numbers. Increasing the accuracy allows a better solution to be found but at the same time increases the total number of points in the parameter space decreasing the convergence. The binary and the binary Gray encodings allow the genetic algorithm to access and operate on individual bits of a gene, instead of the gene as a whole as in the real encoding [110]. For instance, a single-point crossover operation may take place with the crossover point in the middle of a gene in the binary encodings but only between genes in the real encoding. The Gray binary encoding differs from the binary encoding in that consecutive integer numbers differ by only one bit. This difference induces different paths in the genetic algorithm optimization procedure. For instance, mutation of one bit in a gene produces an incremental change in the value of the corresponding physical parameter if it is represented in Gray code, but may lead to a large variation in the case of binary encoding.

The program starts by randomly generating a population of chromosomes in the specified encoding and range of values for each variable. As an example, **FIG. 26b** shows the chromosome of a double-layer grating with its genes corresponding to the physical parameters of the structure illustrated in **FIG. 26a** [65,71]. The chromosome has $(5N_L + 1)$ genes where N_L is the number of layers of the diffractive structure. Each layer is assumed to be a grating with the same period Λ, but with different refractive indices n_H and n_L, thicknesses d, and coordinates (relative to the grating period) of the high-refractive index region of each grating X_L and X_H. To select the refractive indices in each layer, the algorithm generates integer random numbers, which represent pointers to refractive index values in the corresponding input file. Homogeneous layers are generated either when the same refractive index is selected for both regions of the binary grating, or when the fill factor defined as $(X_H - X_L)$ is smaller or greater than the values specified by the user in the input file f_{min} and f_{max}. For $(X_H - X_L) < f_{min}$ the layer is considered as homogeneous with the refractive index n_L, while for $(X_H - X_L) > f_{max}$ the layer is homogeneous with refractive index n_H. Different values of X_L in different layers generate phase-shifted layers. The number of layers N_L is fixed and provided by the user. However the program can analyze structures with fewer layers whenever it selects a layer thickness that is smaller than the minimum layer thickness (from the input file). In this case, the thickness is set to zero and the number of layers decreases by one.

The number sequence forming a population of strings is unique for each run of the program. A feature is included that allows the same number sequence to be generated each time for debugging or reproducibility purposes [110]. In binary representation, each bit of a string has equal probabilities of being set to 0 or 1. In the real encoding the genes are set to a value selected uniformly within the user-specified range.

The population (*i.e.*, the total number of chromosomes generated in the beginning, which is to remain constant after each iteration) is established by the user depending on the dimension of the search space and the length of the chromosome. An increased number of genes and/or a large range of variation for the genes may require a large population for effective optimization. Operating with larger populations, the genetic algorithm is more likely to find the global minimum of the merit function since it

searches more regions of the space simultaneously. However, this is achieved at the expense of an increase in computational time, which imposes a practical limitation on the population size.

The initially generated population is evaluated by calculating a merit function for each chromosome as the deviation between the synthesized value of reflected (or transmitted) zero-order diffraction efficiency and the desired one. The genetic algorithm searches for the global minimum of the following merit function

$$MF = \left[\frac{1}{M} \sum_{i=1}^{M} w_i \left| DE_{GA,i} - DE_{ref,i} \right|^n \right]^{1/n} \tag{5.1}$$

where $DE_{GA,i}$ are zero order reflected (or transmitted) diffraction efficiency values calculated with rigorous-coupled wave analysis for the structure generated by the genetic algorithm, $DE_{ref,i}$ are the reference data points, M is the total number of target values, w_i are the weight factors, and n is the power index of the merit function. The target points represent either a wavelength or an angular dependence of a diffraction order efficiency. Any diffraction order may be selected for optimization, but for the applications of interest to this work concerning only zero-order gratings, the zero-order efficiencies are utilized. The power index of the merit function can take any integer values but in thin-film optics optimization routines the most common value is n = 2. Different values of n, can affect the optimization results due to changes induced in the relative contributions of individual target deviation points $|DE_{GA,i} - DE_{ref,i}|$ to the merit function. For larger values of n higher deviations, will be emphasized and the merit function becomes more sensitive to nonequal deviations forcing the genetic algorithm to find a more uniform approximation to the reference data [96].

Once the merit function has been calculated, the chromosomes are ranked from the best-fit to the least-fit, with the best-fit possessing the lowest merit function. A number of chromosomes are retained while others are replaced by newly generated

chromosomes. The selection mechanism typically used is the tournament selection consisting in retaining the best chromosomes of a population. Other selection mechanisms such as probabilistic tournament (with an associated probability of selecting a chromosome), proportional and stochastic universal selection can also be chosen for use in the optimization procedure [110]. The number of chromosomes replaced is an input parameter to the program and has an important influence on the optimization progress. A high percentage of chromosomes replaced provides more new points for fitness testing which is beneficial in the search procedure, but it will also increase the computation time. It is also possible that a large replacement will cause the elimination of certain chromosomes that, after subsequent crossover and mutation, would have generated the optimum solution. Therefore, several convergence tests need to be performed to establish the optimum population replacement for a specific problem [77].

The new population that replaces the discarded chromosomes is formed by crossover and mutation of the chromosomes that are retained from the old generation [110]. The chromosomes that survive become parents and generate enough chromosomes to maintain the total population constant from one generation to another. The algorithm allows the user to decide whether a string can undergo both crossover and mutation or just one of the two operations.

In the case where either crossover or mutation is carried out, the probability of going towards one or the other operation is decided by a random logical variable that has an associated flip probability (provided in the input file) of returning a logical value "true." A probability of 0.5 corresponds to flipping an unbiased coin. In the case when both mutation and crossover are performed, the random logical value of the flip probability determines whether crossover is executed first followed by a mutation operation or vice versa [110].

Crossover takes place by pairing the chromosomes selected to survive from the old generation into the new one from top to bottom of the list (with best-ranked strings at the top). The crossover operation is performed with a probability defined by the user in the input file. The algorithm has the options of single-point, two-point or uniform

crossover (**FIG. 25**). For the latter type of crossover, the probability of swapping two parent bits (or genes in case of real encoding) called uniform crossover probability, must be specified in the input file.

Mutation takes place with a probability defined by the user in the input file. For binary encoding, mutation is performed by replacing one or more of the bits of a chromosome with its complement. For real encoding, the mutation occurs for one or more genes of a chromosome and can be one of several types: "range," "constant," "uniform," or "Gaussian" [110]. If the mutation is of the "range" type, the gene will be replaced with a number selected with equal probability from the allowed range of variation for the gene. In the other three mutation types the gene g is replaced by $g \pm p \times g$ where the value of p is determined differently for each mutation operator. For constant mutation, p is a constant provided by the user. Uniform mutation occurs when p is selected uniformly from an interval $[0 - M_u]$ where M_u is an input parameter. In the Gaussian type of mutation p is generated by a Gaussian distribution with mean 0 and standard deviation σ given in the input file.

After generating a new pair of chromosomes through crossover and/or mutation, the algorithm performs a verification to determine whether they are different from their parent chromosomes. If the new chromosome is identical to the parent chromosome, the mutation operator is applied to the new chromosome until at least one mutation has occurred.

After evaluation of the newly generated chromosomes and ranking the new generation, the process of selection, crossover, and mutation is repeated until the stopping criteria is met. This can be determined by a fixed number of iterations, no change in the best string after a number of iterations, or when certain fraction of the population has the same merit function [110]. The genetic algorithm also has the restarting option by which the best string is kept and all others are generated as mutants of the best string. The number of iterations between restarting operations is defined by the user.

The program prints the best N_{out} chromosomes and their corresponding merit functions in the output file, where N_{out} is an input parameter. By printing a number of the

top chromosomes, the user can asses the distribution of solutions and hence the degree of convergence of the algorithm. A large dispersion in the gene values of the final chromosomes indicates that the algorithm has not yet converged and changes need to be made in the input parameters of a future run. Typical changes would be to try more iterations, larger populations, or impose more constraints according to a priori knowledge about the physics of the problem [111]. However changes in other genetic algorithm input parameters can also improve the convergence and the effectiveness of the optimization.

Convergence tests

The genetic algorithm developed in this work has a general applicability. The genetic operators and optimization procedure can be utilized in any optimization task involving multilayer structures containing gratings and homogeneous layers, with minor modifications pertaining to the encoding and decoding of the chromosomes. The merit function evaluation subroutines can be applied to optimization of any structure that can be modeled with the rigorous coupled-wave analysis.

However, the optimum set of the program-input parameters may be problem specific due to the dimension of the solution space and the particular variation of the merit function in the parameter space. To determine the influence of the input parameters on the optimization procedure and final result, and to find some guidelines for selecting the appropriate set of input parameters for a specific application, it is important to study the evolution of the optimization process (*i.e.*, the convergence) for various starting conditions [65,77].

In this section, the convergence of the merit function is studied as a function of key genetic-algorithm parameters such as the population replaced at each iteration, mutation probability, type of encoding, number of generations, population size, for the same problem. In all tests discussed here, the program is required to design a single-layer guided-mode resonance reflection filter with the response specified in the input file by the spectral dependence of the zero-order reflection diffraction efficiency. This reference data is generated with the rigorous coupled-wave theory for a single-layer grating with

the following physical parameters: grating period $\Lambda = 314$ nm, thickness d = 134 nm, fill factor f = 0.5, refractive indices of the grating: $n_H = 2.1$ and $n_L = 2.0$, refractive indices of the cover and substrate: $n_C = 1.0$ and $n_S = 1.52$, and normally incident, TE polarized plane wave. The optimization is performed in terms of the layer thickness, fill factor and refractive indices of the grating over a wavelength range 0.546 – 0.554 μm. The grating period is fixed at the value $\Lambda = 314$ nm, the cover and substrate refractive indices have constant values of $n_C = 1.0$ and $n_S = 1.52$, respectively, and the incident angle is set at $\theta = 0°$. The allowed range for fill factor optimization is between 0.1 - 0.9 and the thickness range is 50 - 350 nm. Throughout the tests the algorithm uses the same set of 13 refractive indices with values from 1.3 - 2.5 in increments of 0.1. The materials are assumed to be lossless although the program can handle lossy grating structures as well.

Comparing the merit function values for tests with the three different types of crossover, it was found that multiple crossover yields the best results in comparison with the single-point and two-point crossover operators. The crossover probability was maintained at 0.8 and the uniform crossover probability was 0.5 for all tests performed. Other genetic algorithm parameters kept constant for all tests were the flip probability equal to 0.5, the tournament selection type, and maximum iteration as the stopping criterion.

The tests performed with populations of 500 and 1000 chromosomes indicate that although fast convergence and low merit function values are also possible with a smaller number of chromosomes, for certain values of the genetic algorithm parameters, the larger population is generally expected to yield lower merit functions, for all other parameters being constant. However, in some cases the increased computation time for larger populations may not be rewarded by a substantial decrease in merit function and an optimum population has to be determined for a typical chromosome length and search space. It has been observed that larger populations also provide the algorithm with less sensitivity to the other input parameters, and therefore fewer trials are required to determine the optimum set of genetic algorithm parameters.

A more detailed investigation of the convergence sensitivity to genetic algorithm parameters was performed for the population replacement and the mutation probability, contrasting real versus binary and Gray encoding performance [77]. Hence, the population size was fixed at 500, the crossover type was uniform with probability fixed at 0.8, and the number of generations and mutation probability were varied. For the real encoding the mutation type was chosen to be Gaussian with standard deviation $\sigma = 0.1$. In all cases the newly created strings were specified to undergo both crossover and mutation. When using binary or Gray encoding, 10 bits were chosen to represent the thickness, 10 bits to represent the fill factor, and 4 bits to represent the pointer to the set of materials. In this case uniform crossover with probability 0.8 was chosen. The number of generations was taken to be 400, to ensure that convergence had been reached.

After performing at least four runs for the same set-up, with different random number sequences in the genetic algorithm functions to seed the population, in order to construct a statistically significant sample of the outcomesit was noted that the real encoding produced smaller merit functions, therefore indicating that for this problem it is better suited than the others. Overall, the sensitivity to replacement values over the range 50 – 250 chromosomes was not very strong considering the total distribution of results. In the case of real encoding, on average, the lowest merit function was obtained for a replacement value of 150. In the case of binary and Gray encodings, slightly smaller merit functions were achieved for the replacement value of 250. However, this replacement also produced a large spread of the residuals indicating that a poor solution can obtained as well as a good one.

Turning to the behavior of the residual as a function of increasing values of mutation probability, for the binary and Gray encoding, a replacement value of 50 was used, whereas for the real encoding the value was 200. Otherwise, the same genetic algorithm parameters as discussed above were retained. The binary and Gray encoding results were not very sensitive to the value of mutation probability. However, on average, the binary encoding favored lower mutation probabilities than the real encoding, which benefits from an increase in the value of this parameter to 1. In the real encoding, the mutation operator was applied to the gene as a whole, which, for the structures

studied here, are represented by 4 numbers. In the binary encoding, mutation took place at the level of individual bits (*e.g.*, for a representation of the filter parameters (d, f, n_H, n_L) with (10, 10, 4, 4) bits, the mutation is applied individually to all of the 28 bits of the chromosome with the specified mutation probability).

Therefore, for the mutation to be active in the search procedure with the real encoding, the mutation probability must be higher than in the binary case where a low probability is compensated by the larger number of elements (bits) to which it is applied. The differences observed between the binary and the real encodings were also due to the different manner in which mutation was carried out. In the real encoding, the Gaussian mutation did not change the gene value by an arbitrarily large amount as in the binary case, but applied a random change in a limited range (determined by σ) around the value of the gene.

Thus, in the real encoding, mutation performed a more localized search alongside the crossover operator before shifting to a different region of space and was able to find a global minimum with greater accuracy. It was found that mutation probabilities of 1 were detrimental in the binary encoding due to the rapid changes occurring in the best chromosomes, which prevented the fine tuning performed by crossover. On average, very low mutation probabilities (0.001) produced equally poor results by precluding the algorithm from exploring new regions in the parameter space.

Turning to the convergence history for three different values of number of chromosomes replaced at each generation (*i.e.*, 50, 150 and 250), when increasing this number, note that the value of the merit function at convergence was reached after a decreasing number of generations. While the computational burden increased with the increase of the chromosomes replaced per generation, this was offset by the ability to reach convergence in fewer generations.

Therefore, the total amount of calculations necessary to reach convergence was similar in all cases. Note that the real encoding generally provided lower merit functions than the binary and Gray encodings. The high sensitivity of the guided-mode resonances to structural parameters was equivalent to a rugged search space for the genetic algorithm

with multiple and narrow local extrema. Therefore, the improved fine-tuning performed by the mutation operator in the real encoding lead to superior convergence results.

The number of reference reflectance vs. wavelength points has been found to significantly influence the final result of the optimization process. It is well known from homogeneous thin-film optimization techniques that more "targets" typically lead to improved results due to the additional information supplied by the user [94].

The number of data points was an even more critical parameter in the design of grating devices that exhibit sharp variations in the reflectance and transmittance spectral dependence. In this case, the global minima of the merit function can not be reached if insufficient target data is provided. In reference 112, for instance, the micro-genetic algorithm uses only one reference point with zero-order reflectance equal to 1 at the wavelength $\lambda = 1.0$. The resulting structure exhibits an almost 100% peak at $\lambda = 1.0$ but other high-efficiency peaks are also present in the proximity of the desired peak thus limiting the filter range. The side peaks can be eliminated in the optimization process by providing the merit function with more reference reflectance points. In the present work, between 40-80 reference data points were used. The major drawback of the increased number of target data was the increased computation time.

The distribution of reference reflectance points in the spectral range of interest was also important in the search for an optimum design. In the case of the rapidly varying reflectance characteristics studied here, it was advantageous to use unequally spaced data points with more reflectance values in the resonance spectral region and less in the sidebands. Utilizing this type of distribution, the algorithm was able in all tests to find a resonance and typically within $\pm$ 0.1 nm of the reference reflectance peak.

A different method to emphasize some reference points over others is to introduce different weight factors. Increased values of the weight factors in some reference points will raise the accuracy in these spectral regions at the expense of a larger target deviation in wavelength regions considered of lesser importance.

REFERENCES

The following references, to the extent that they provide exemplary procedural or other details supplementary to those set forth herein, are specifically incorporated herein by reference.

[65] S. Tibuleac, D. Shin, R. Magnusson, and C. Zuffada, "Guided-mode resonance filters generated with genetic algorithms," Proceedings of the Topical Meeting on Diffractive Optics and Micro-Optics, Kailua-Kona, Hawaii, June 1998, Conference Proceedings, vol. 10, pp. 24-26, 1998.

[71] S. Tibuleac, D. Shin, R. Magnusson, and C. Zuffada, "Design of reflection and transmission guided-mode resonance filters with genetic algorithms," Optical Society of America Annual Meeting, Baltimore, Maryland, October 1998, Conference Proceedings, p. 128, 1998.

[77] C. Zuffada, D. Levine, and S. Tibuleac, "Designing dielectric grating filters with PGAPACK," Electromagnetic system design using evolutionary optimization: genetic algorithms, edited by Y. Rahmat-Samii and E. Michielssen, John Wiley and Sons, 1999.

[84] T. K. Gaylord and M. G. Moharam, "Analysis and applications of optical diffraction by gratings," Proc. IEEE, vol. 73, pp. 894-937, May 1985.

[85] M. G. Moharam, E. B. Grann, D. A. Pommet, and T. K. Gaylord, "Formulation for stable and efficient implementation of the rigorous coupled-wave analysis of binary gratings," J. Opt. Soc. Am. A, vol. 12, pp. 1068-1076, May 1995.

[86] M. G. Moharam, D. A. Pommet, E. B. Grann, and T. K. Gaylord, "Stable implementation of the rigorous coupled-wave analysis for surface-relief gratings: enhanced transmittance matrix approach," J. Opt. Soc. Am. A, vol. 12, pp. 1077-1086, May 1995.

[94] TFCalc manual, Thin Film Design Software for Windows, Version 3.0, Software Spectra, Inc., 1995.

[96] Sh. A. Furman, and A. V. Tikhonravov, Basics of Optics of Multilayer Systems, Editions Frontieres, Paris, 1992.

[104] D. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, Massachusetts, 1989.

[105] B. P. Buckles and F. E. Petry, Genetic Algorithms, IEEE Computer Society Press, Los Alamitos, California, 1992.

[106] L. Davis, Ed., Genetic Algorithms and Simulated Annealing, Pitman, London, 1987.

[110] D. Levine, "Users guide to the PGAPack parallel genetic algorithm library," Argonne National Laboratory, ANL 95/18, January 1996.

[111] R. L. Haupt, "An introduction to genetic algorithms for electromagnetics," IEEE Antennas and Propagation Magazine, vol. 37, pp. 7-15, April 1995.

[112] E. G. Johnson and M. A. G. Abushagur, "Microgenetic-algorithm optimization methods applied to dielectric gratings," J. Opt.